Challenges of large-scale data synchronization

Petr Kuznetsov
ACES, Télécom Paris, IP Paris
Gross Buch
Fault-tolerant state machine replication

- Paxos [Lamport 91]
- Byzantine (arbitrary) faults: PBFT [Castro-Liskov 1999]

  - Partial synchrony
  - Byzantine quorum systems
  - f < N/3 replicas may be faulty
Challenge: open environment:

- **Permissionless**: no static membership
- **No identities**: public keys
- **Sybil attack**: any participant subset can be adversarial

Classical *(partially synchronous quorum-based)* protocols do not work!
Sybil-resistant consistency: PoW “consensus”

- **Synchrony**: slow down updates
- Solve a difficult puzzle before updating (PoW)
- Throughput low by design

*Is consensus necessary?*
Consensus

Processes *propose* values and must *agree* on a common decision value so that the decided value is a proposed value of some process.
Why consensus is interesting?

Because it is universal!

- A key to implement a generic fault-tolerant service (*replicated state machine* or *blockchain*)

Expensive and cumbersome

Is consensus necessary for a cryptocurrency (asset transfer)?

Guerraoui et al. The consensus number of cryptocurrency. PODC 2019
Commutativity and causality

- T0: $100 from Alice to Carole
- T1: $100 from Bob to Alice
- T2: $100 from Drake to Alice

T0 \textit{causally depends} on T1 (not enough funds otherwise)
T1 and T2 \textit{commute} (T0 succeeds regardless of the order)
Consensus-less cryptocurrency

- Each transfer relates to its causal past (incoming/outgoing transactions)
- Make sure that a faulty account holder cannot lie about its causal past
  ✓ Source-order: messages by the same source are delivered in the same order

Collins et al. Online payments by merely broadcasting messages [DSN20]
Total order vs. partial order

- Consensus = total order
  - Participants learn an ordered sequence

- Lattice agreement = partial order
  - Participants learn a partially ordered sequence
Lattice Agreement on \((L, \sqsubseteq, \sqcup)\)

- **L** – set of values, \(\sqsubseteq\) - partial order, \(\sqcup\) - join operator
  - **Comparability**: all learned values are comparable
  - **Validity**: every learned value is a join of proposed values
  - **Liveness**: every value proposed by a correct process eventually appears in a learned value

Allows for efficient asynchronous implementations [FRR+, 2012]

Perfect fit for asynchronous reconfiguration [OPODIS19, DISC20]
Permissionless asset transfer?


Kuznetsov, Pigolet, Ponomarev, Tonkikh. Permissionless and asynchronous asset transfer. DISC’21
Permissionless and asynchronous asset transfer

Idea:
- Use weighted (stake-based) quorums
- A transaction is accepted if validated by $>2/3$ of stake

Solution:
- Treat stake distribution as a configuration
- A transaction is a reconfiguration call
- Reconfigurable Lattice Agreement as a building block

Permissionless and asynchronous asset transfer
[Kuznetsov et al., DISC 2021]
Strong consistency of data in an open system: a hard problem in a hard model?

- **Relax the problem**
  - Asset transfer (LADT [OPODIS19]) instead of blockchain [PODC 2019, DSN 2020, DISC 2021]
  - Multiple spending [Bezerra et al., PODC 2022]
  - Accountability vs. fault-tolerance [Freitas et al., OPODIS 2021]

- **Strengthen the model**
  - (Eventual) synchrony
  - Stake assumptions
  - Some trust (federated quorums)
TrustShare 2021: Innovation Chair

- **Reconfigurable systems**
  - The set of participants can be (actively) reconfigured without consensus
    [OPODIS 2019, DISC 2020]

- **Randomness in blockchain protocols**
  - Leader election and sortition in a blockchain protocol
    [OPODIS 2021], approximate random coin
    [DISC 2022]

- **Accountability**  [SOSP 2007, OPODIS 2009, PODC 2021, OPODIS 2021]
  - Detect misbehavior rather than anticipate it

- **Asynchronous cryptocurrency**  [PODC 2019, DISC 2019, DSN 2020, DISC 2021]
  - Use stake for permissionless asset exchange

- **Decentralized trust assumptions**  [PODC 2022]
  - Double spending under control

- **Security and privacy** in sharing data, reconciling blockchains, coding for communication-efficiency and more…
Merci!
Accountability and asynchronous reconfiguration

How to reconfigure?

Consensus-based:
- RAMBO [Gilbert et al., 2010]
- Casper [Buterin-Griffith, 2017]
- Fairledger [Lev-Avirt et al., 2019]
- LLB [Ranchal-Pedrosa & Gramoli, 2020]

Asynchronous:
- Lattice-agreement instead of consensus [Kuznetsov et al., 2019]

Accountable and reconfigurable lattice agreement [Freitas et al., OPODIS 2021]